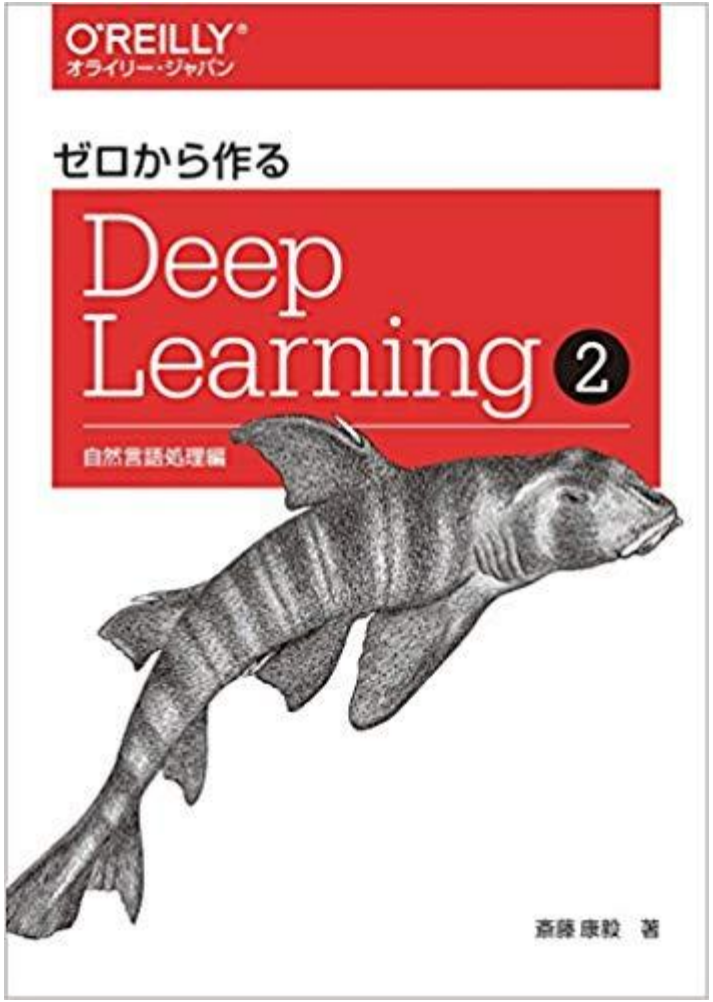
自然言語処理入門

Natural Language Processing

本日の教材:

ゼロから作る Deep Learning 2
自然言語処理編

(4000円くらいする)



目次:

1. ニューラルネットワークの復習
2. 自然言語と単語の分散表現
3. word2vec
4. word2vec の高速化
5. リカレントニューラルネットワーク (RNN)
6. ゲート付き RNN
    - LSTM
7. RNN による文章生成
    - seq2seq
8. Attention
    - Transformer, BERT

時間が余ったら...

- word2vec の学習済みモデル (pre-trained model) を使って遊ぶ
- WordNet で遊ぶ

2章　自然言語と単語の分散表現

自然言語処理

本質的問題:

コンピュータに自然言語を理解させる

自然言語:

私たちが普段使っていることば

応用例:

検索エンジン，機械翻訳，質問応答システム，かな漢字変換，文章の自動要約，感情分析，など

単語の意味

私たちの言葉は「文字」によって構成される

私たちの言葉の意味は「単語」によって構成される

＞単語は意味の最小単位

自然言語をコンピュータに理解させる
＝「単語の意味」を理解させる

「単語の意味」を理解させる手法

- **シソーラスによる手法**
  - 人の手によって作られたシソーラス（類語辞書）を利用する手法
- **カウントベースの手法**
  - 統計情報から単語を表現する手法
- **推論ベースの手法** (3章)
  - ニューラルネットワークを用いた手法

シソーラスによる手法

人の手によって作られたシソーラス（類語辞書）を利用する方法

[背景]

単語の意味を表現する方法の例：

「広辞苑」のように一つひとつの単語の意味を説明していく

自動車 - 車輪を取り付けてそれによって進むようになっている乗り物や…

＞単語を定義すれば，コンピュータも単語の意味を理解できるかも？

シソーラスによる手法

自然言語処理のアプローチ:

「広辞苑」のような一般的な辞書ではなく,

シソーラスと呼ばれる辞書が多く使われてきた

シソーラス:

類語辞書であり,「同じ意味の単語（同義語）」や「意味の似た単語（類義語）」が同じグループに分類されている

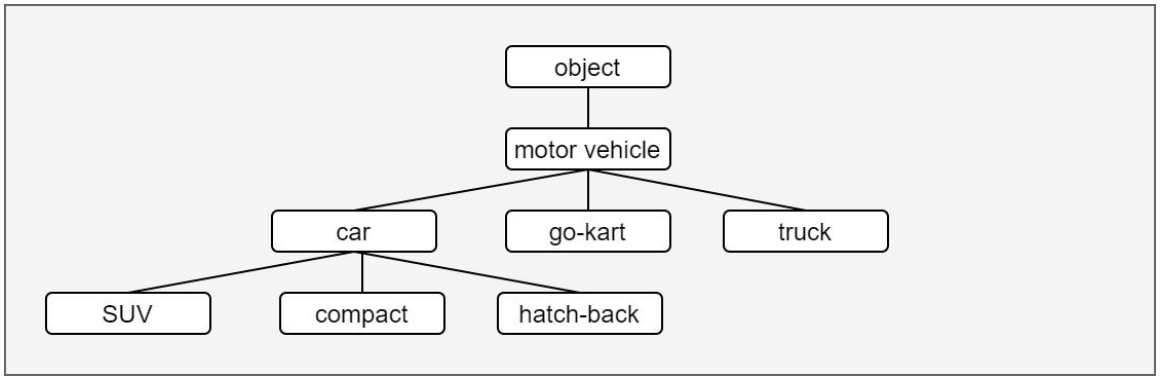
car = auto automobile machine motorcar

シソーラスによる手法

自然言語処理で利用されるシソーラス:

単語の間で,「上位と下位」,「全体と部分」などの，より細かい関連性が定義されている

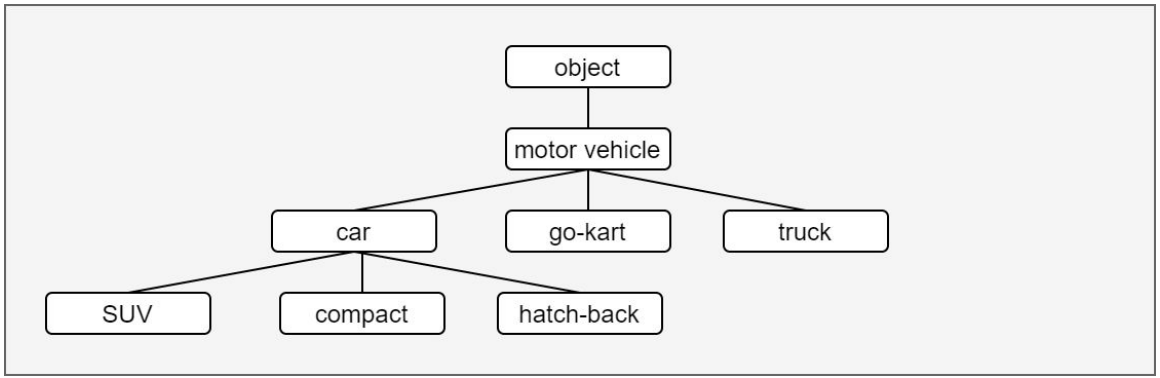
＞単語ネットワーク



シソーラスによる手法

単語のネットワークを用いて，コンピュータに単語の意味を間接的に授けられる

この知識をもとに，有用な処理をコンピュータに行わせることが可能



WordNet

WordNet:

自然言語処理の分野において，最も有名なシソーラス

1985年にプリンストン大学で開発がスタート

日本語版も存在する

＞python で動かせるので，時間が余ればやりましょう

シソーラスの問題点

シソーラスによる手法は，言い換えると，

「単語の意味を人手によって定義する手法」

問題点：

1. 時代の変化に対応するのが困難
2. 人の作業コストが高い
3. 単語の細かなニュアンスを表現できない

シソーラスの問題点の解決

シソーラスによる手法 =「単語の意味を人手によって定義する手法」

問題の解決:

- カウントベースの手法
- 推論ベースの手法

＞大量のテキストデータから自動的に「単語の意味」を抽出

＞＞人手はほとんど必要ない

カウントベースの手法

カウントベースの手法では，コーパスを利用する

コーパス：

大量のテキストデータ，自然言語処理の研究やアプリケーションのために目的をもって収集されたテキストデータ

例：wikipedia, google news, twitter, amazon review, ...

カウントベースの手法

コーパスに含まれる文章は，人の手によって書かれたもの

＞コーパスには自然言語に対する人の”知識”がたくさん含まれている

カウントベースの手法の目標:

人の知識が詰まったコーパスから，自動的に，効率よく，そのエッセンス（特徴量）を抽出すること

単語の分散表現

分散表現:

単語を (単語の意味を的確に捉えた) ベクトル表現で表す

イメージ: 色に対するRGB値 (赤 - (R,G,B) = (255, 0, 0))

利点:

正確に単語を指定できる

単語どうしの関連性を容易に判断できる

定量化できる (例えば，数値計算可能 - [King - Man + Woman = Queen])

分布仮説

自然言語処理の歴史において，単語をベクトルで表す研究はたくさんある

＞重要な手法のほとんどすべてがひとつのシンプルなアイデアに基づいている

単語の意味は，周囲の単語によって形成される

これを「分布仮説 (distributional hypothesis)」と呼ぶ

分布仮説の意味:

単語自体には意味がなく，その単語の「コンテキスト (文脈)」によって，単語の意味が形成される

分布仮説

分布仮説の直感的な理解:

“I drink beer.” “I drink wine.” - drink の近くには飲み物が現れやすい

“I guzzle beer.” “I guzzle wine.” - guzzle という単語は drink と同じような文脈で使われる

(英語のセンター試験の単語の意味推定に似てる...)

＞drink と guzzle は近い意味の単語

カウントベースの手法も推論ベースの手法も分布仮説がベース

コンテキスト

コンテキストとは？：

you say goodbye and i say hello.

ウィンドウサイズが2のコンテキストの例

“goodbye” という単語に着目した時，左右の2単語をコンテキストに利用

共起行列

分布仮説に基づいて，単語のベクトル表現を表す最もシンプルな方法

＞周囲の単語を「カウント」すること

よって，カウントベースの手法 (count-based method) と呼ばれる

(統計的手法を呼ばれる場合も)

共起行列

you say goodbye and i say hello.

というコーパスに対する共起行列 (window size = 1) は以下

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
goodbye	0	1	0	1	0	0	0
and	0	0	1	0	1	0	0
i	0	1	0	1	0	0	0
hello	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0

当たり前だが，共起行列は対称行列

カウントベースの手法で得られる分散表現

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
goodbye	0	1	0	1	0	0	0
and	0	0	1	0	1	0	0
i	0	1	0	1	0	0	0
hello	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0

カウントベースの手法はこの共起行列をもとに分散表現を得る - 行 (or 列) を抽出

例：you の分散表現 - [0, 1, 0, 0, 0, 0, 0]

say の分散表現 - [1, 0, 1, 0, 1, 1, 0]

カウントベースの手法で得られる分散表現

- 実際に学習に用いられるコーパスはもっとはるかに大きい
  - 大きくないと精度が出ない
  - 数億語，数十億語規模
- よって，共起行列もはるかに大きくなる
  - 共起行列のサイズは 語彙数(vocabulary size) x 語彙数
  - 数万語 x 数万語くらいの規模 (数十万規模のものも)
- つまり，分散表現も数万次元になる

カウントベースの手法で得られる分散表現

- あまり大きいと計算コストなどの観点から利用しにくい
  - 良く用いられるのは 300 次元程度
- よって，行列の次元削減 (dimensionality reduction) を行う
  - 共起行列はスパース (0 ばっか)
  - 行列を 語彙数 x (300) に圧縮したい
  - 特徴量を損なわずに次元削減を行いたい
    - 特異値分解 (Singular Value Decomposition: SVD)

3章　word2vec

シンプルな”word2vec”を実装してみよう！

カウントベースの手法の問題点

問題:

大規模なコーパスを扱う場合に発生

英語の語彙数は100万を超える (全ての単語を扱おうと思ったら)

つまり，共起行列は 100万 x 100万

これを SVD で次元削減するのは現実的でない

SVD は n x n の行列に対して，O(n^3) の計算コスト

カウントベースの手法の問題点

カウントベースの手法:

学習データを一度にまとめて処理する

推論ベースの手法:

学習データの一部を使って逐次的に(= ミニバッチで)学習する

＞データを小分けにして学習できるので計算量が膨大にならない

　並列計算が容易にできる = 学習の高速化が容易

推論ベースの手法

概要:

推論ベースの手法では，「推論」することが主な作業

you ? goodbye and I say hello.

「?」にどのような単語が出現するのかを推測する作業

このような推論問題を繰り返し解くことで，単語の出現パターンを学習する

推論ベースの手法

概要:

推論ベースの手法では，何らかのモデルが登場する

word2vec では，そのモデルにニューラルネットワーク (NN) を用いる

モデルはコンテキスト情報を受け取り，（出現しうるであろう）各単語の出現する確率を出力する

正しい推測ができるようにコーパスを使ってモデルを学習

学習の結果，分散表現が得られる

分布仮説

自然言語処理の歴史において，単語をベクトルで表す研究はたくさんある

＞重要な手法のほとんどすべてがひとつのシンプルなアイデアに基づいている

単語の意味は，周囲の単語によって形成される

これを「分布仮説 (distributional hypothesis)」と呼ぶ

分布仮説の意味:

単語自体には意味がなく，その単語の「コンテキスト (文脈)」によって，単語の意味が形成される

NN における単語の処理方法

NN は “you” や “say” などの単語をそのままでは処理できない

NN で単語を処理するには，それを「固定長のベクトル」に変換する必要がある

その方法のひとつは，単語を one-hot ベクトル に変換すること

one-hot ベクトル：

[1, 0, 0] のようなベクトルの要素の中でひとつだけが1で，残り全てが0であるようなベクトル

one-hot vector

You say goodbye and I say hello.

という1文をコーパスとする（2章と同じ）

単語	単語ID	one-hot vector
you	0	(1,0,0,0,0,0,0)
goodbye	2	(0,0,1,0,0,0,0)

※単語ID は単語の出現順に0から振られるとする

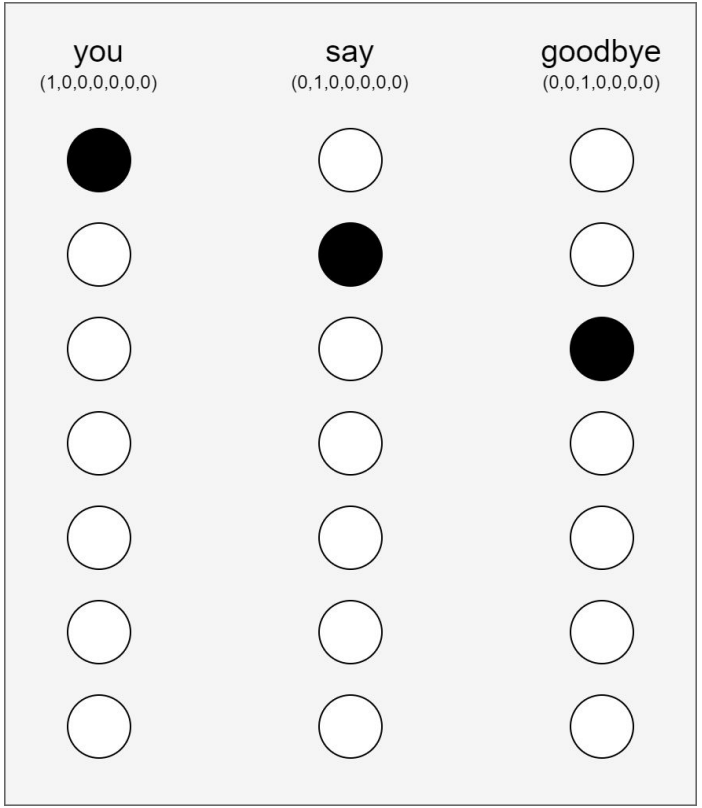
目的:

単語を固定長のベクトルに変換してしまえば，NN の入力層のニューロン数を固定できる

one-hot vector

目的:

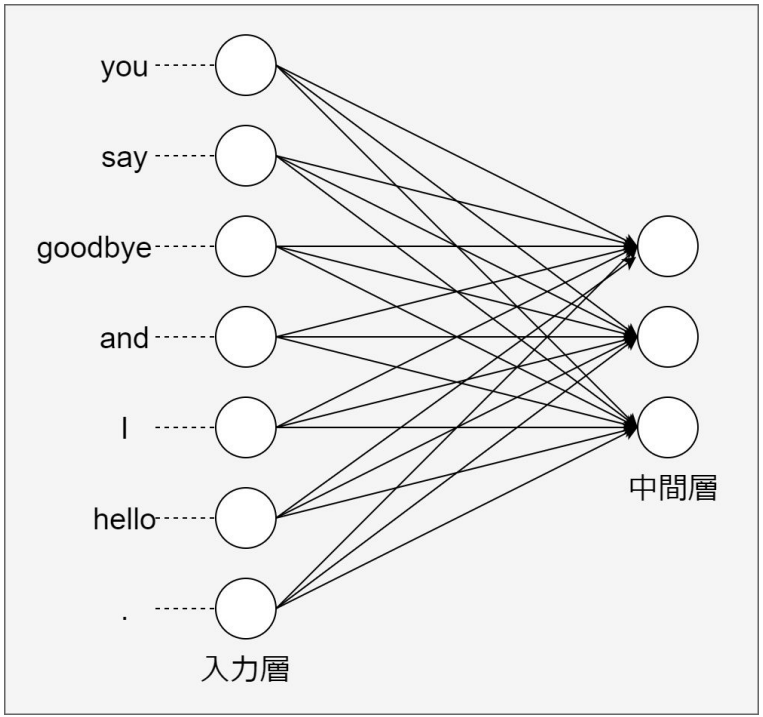
単語を固定長のベクトルに変換してしまえば，NN の入力層のニューロン数を固定できる



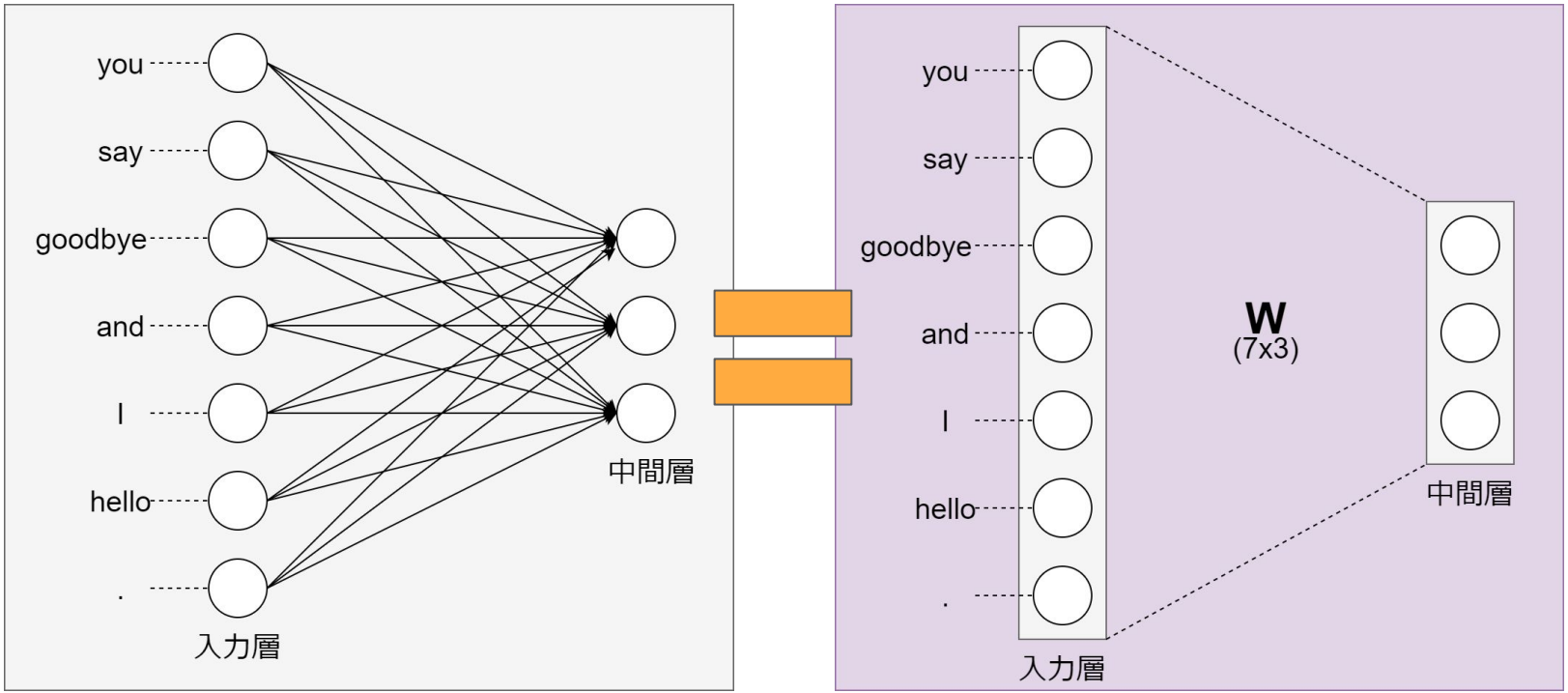
NN における単語の処理方法

one-hot ベクトルで表された一つの単語に対して，全結合層で変換する場合

矢印には重みが存在し，入力層のニューロンとの重み付き和が中間層のニューロンになる



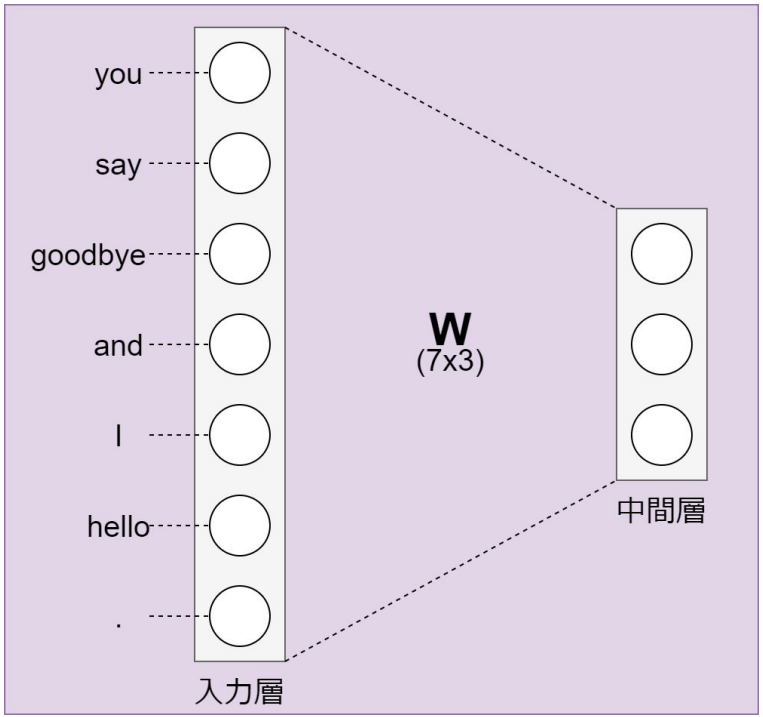
NN における単語の処理方法



全結合層による変換

```
import numpy as np
# 入力
c = np.array([[1, 0, 0, 0, 0, 0, 0]])
# 重み
W = np.random.randn(7, 3)
# 中間ノード
h = np.dot(c, W)
print(h)
```

c は one-hot vector なので, h は W の1つ目の行ベクトルを抜く出すことと同じ



シンプルな word2vec

word2vec を実装してみる

- word2vec
  - continuous bag-of-words (CBOW) モデル
    - 複数の単語（コンテキスト）からひとつの単語（ターゲット）を推測する
  - skip-gram モデル
    - ひとつの単語（ターゲット）から複数の単語（コンテキスト）を推測する

CBOW モデルの推論処理

CBOW モデルは，コンテキストからターゲットを推測することを目的としたNN

ターゲット:

中央の単語

コンテキスト:

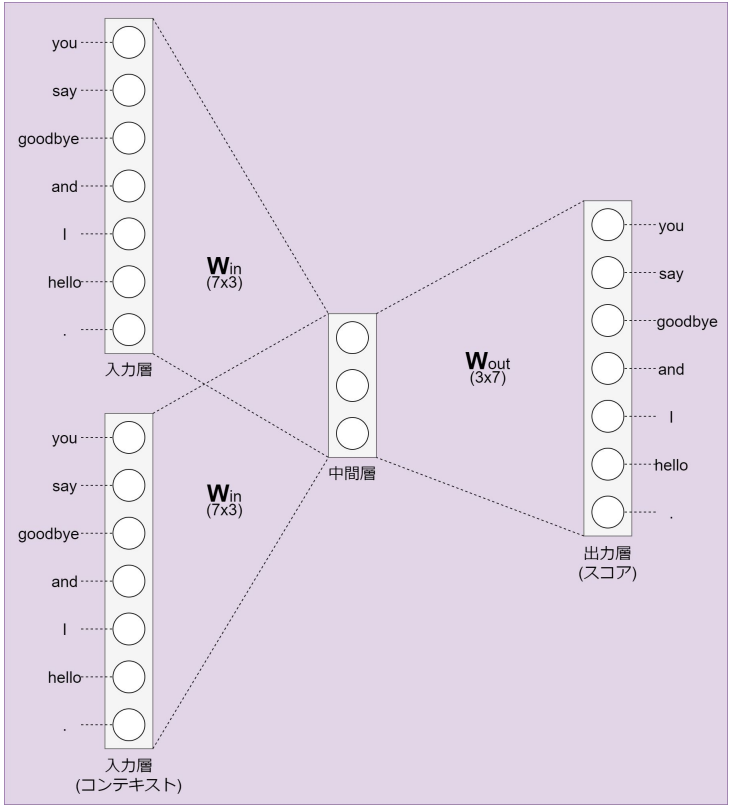
その周囲の単語

CBOW モデルのネットワーク構造

ここではコンテキストとして2つの単語を考えている(window size = 1)ため，入力層が2つ存在している

もしコンテキストして N 個の単語を扱う場合は，入力層は N 個になる

中間層にあるニューロンは，各入力層の全結合による変換後の値が「平均」されたもの



CBOW モデルのネットワーク構造

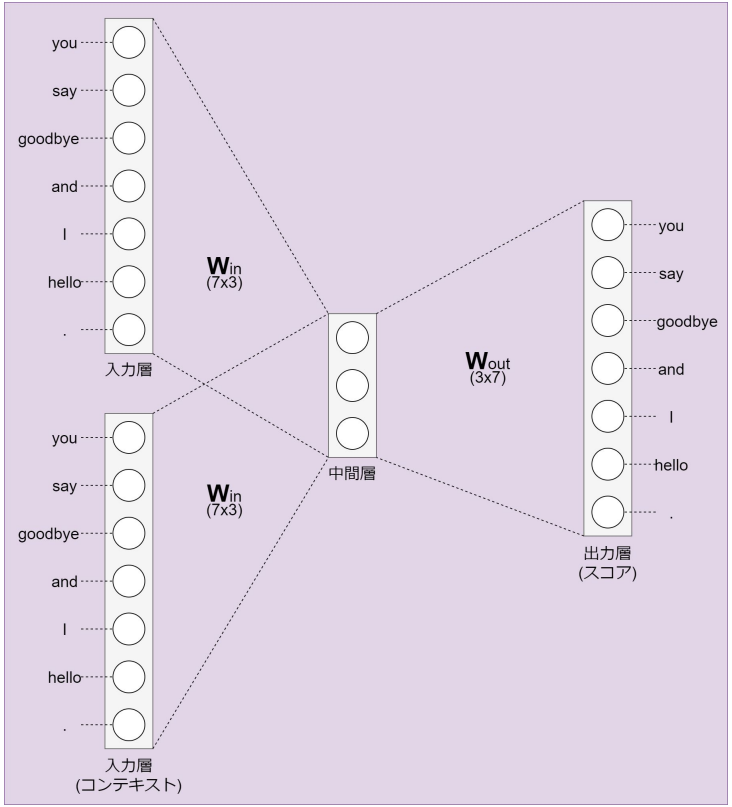
出力層のニューロンは各単語に対応している

出力層のニューロンは各単語の「スコア」であり，その値が高いほど，対応する単語の出現確率も高くなる

スコアを Softmax 関数に適用することで「確率」が得られる

cf. softmax とは？- 合計で1(100%)になるようにノードの値を調整

https://qiita.com/Hatomugi/items/158b8656da5cc4cce0fe

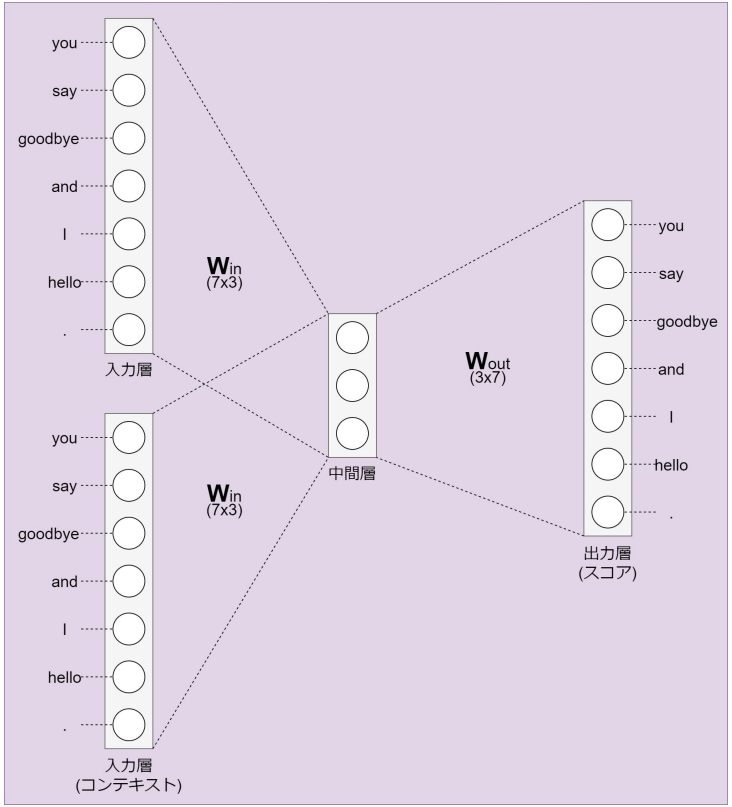


CBOW モデルの分散表現

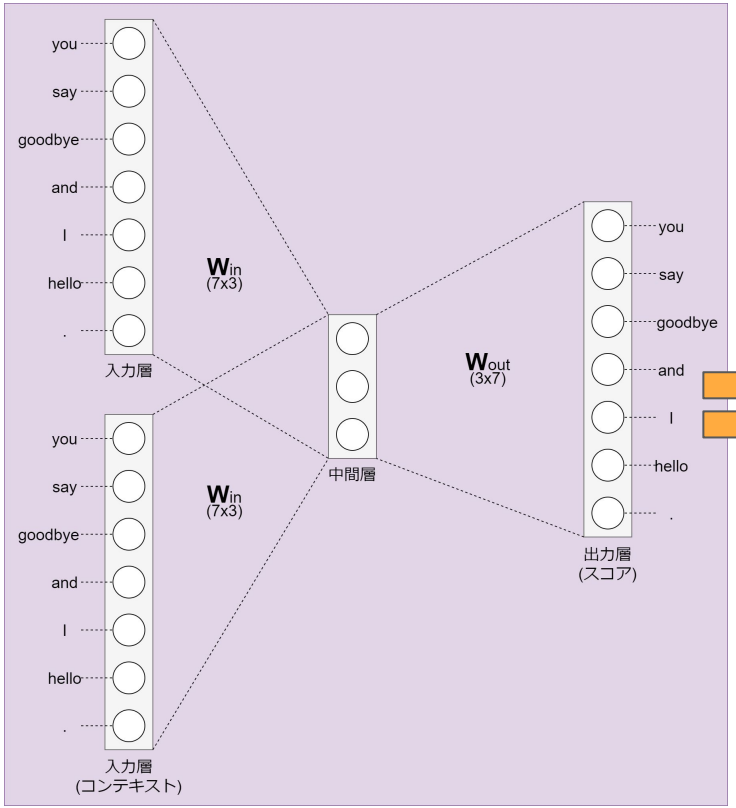
W_{in} が単語の分散表現の正体

学習を重ねるごとに，コンテキストから出現する単語をうまく推測できるように重みが更新される

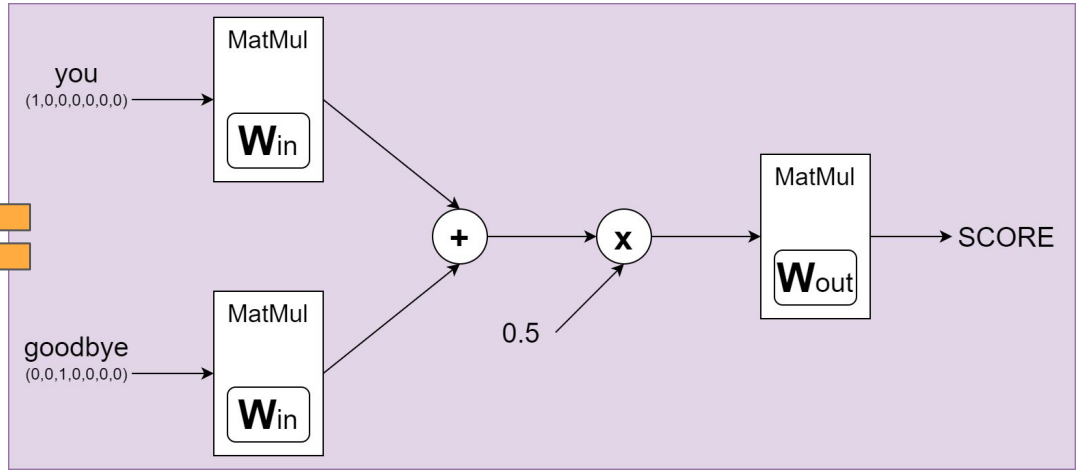
＞各単語の分散表現が更新される



CBOW モデルのネットワーク構成



ニューロン視点



レイヤ視点

CBOW モデルのネットワーク構成

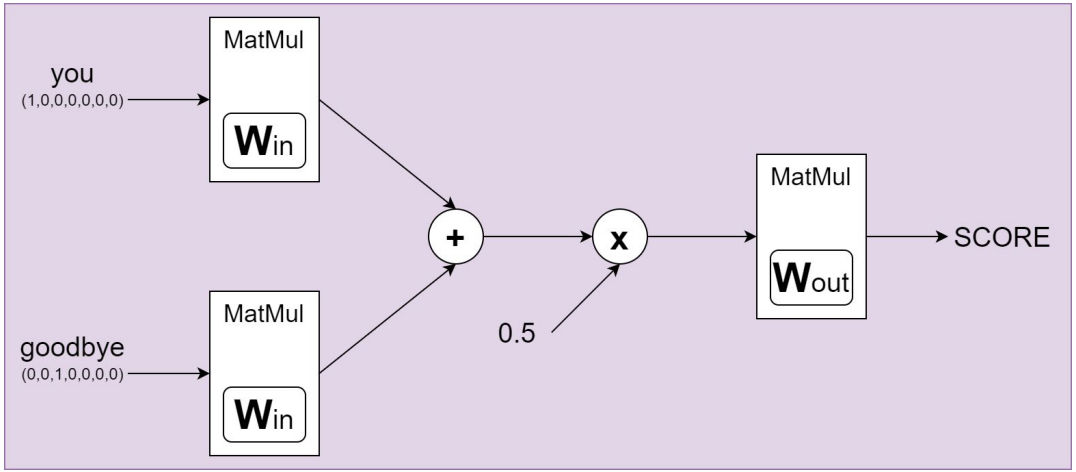
このネットワークを python で書いてみよう

https://github.com/oreilly-japan/deep-learning-from-scratch-2

> ch03/cbow_predict.py

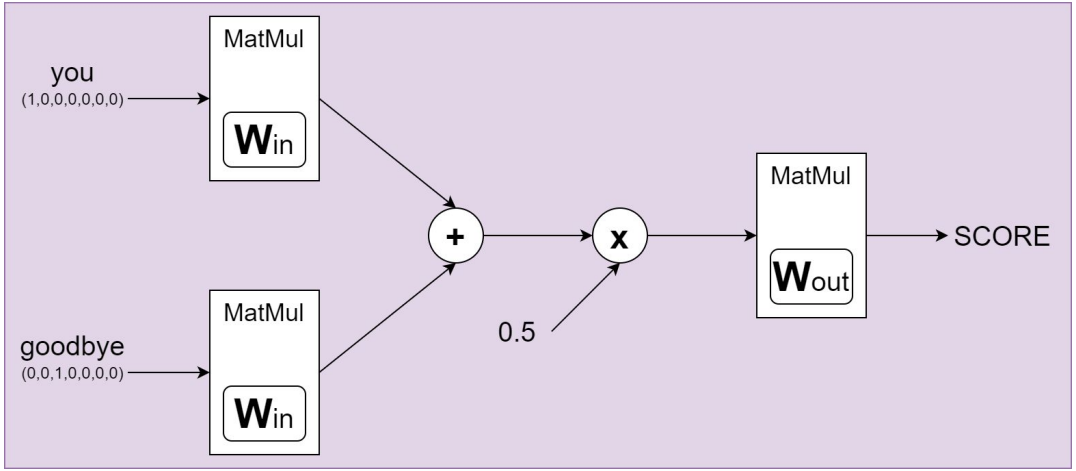
※MutMul ノード:

行列の積を行うノード



CBOW モデルのネットワーク構成

```
# coding: utf-8
import sys
sys.path.append('..')
import numpy as np
from common.layers import MatMul
# サンプルのコンテキストデータ
c0 = np.array([[1, 0, 0, 0, 0, 0, 0]])
c1 = np.array([[0, 0, 1, 0, 0, 0, 0]])
# 重みの初期化
W_in = np.random.randn(7, 3)
W_out = np.random.randn(3, 7)
# レイヤの生成
in_layer0 = MatMul(W_in)
in_layer1 = MatMul(W_in)
out_layer = MatMul(W_out)
# 順伝搬
h0 = in_layer0.forward(c0)
h1 = in_layer1.forward(c1)
h = 0.5 * (h0 + h1)
s = out_layer.forward(h)
print(s)
```



CBOW モデルの学習

you ? goodbye and I say hello.

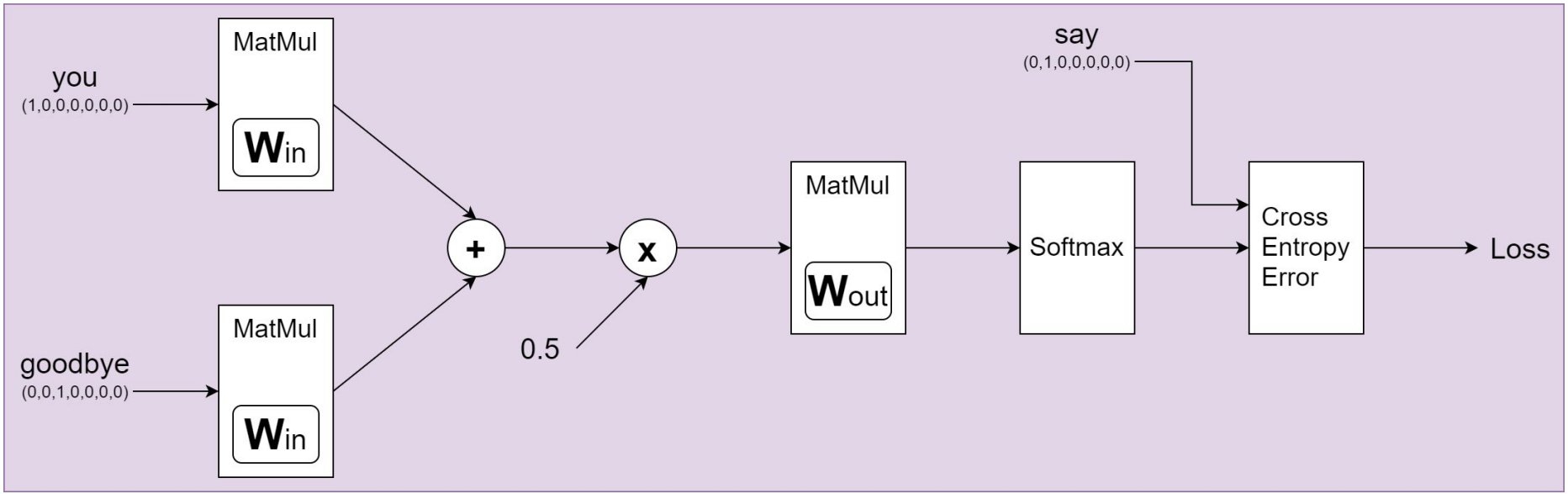
「?」に相当するのは，“say” である.

コンテキストが “you” と “goodbye” の時，“say” の確率が最も高くなるように NN の学習を行う

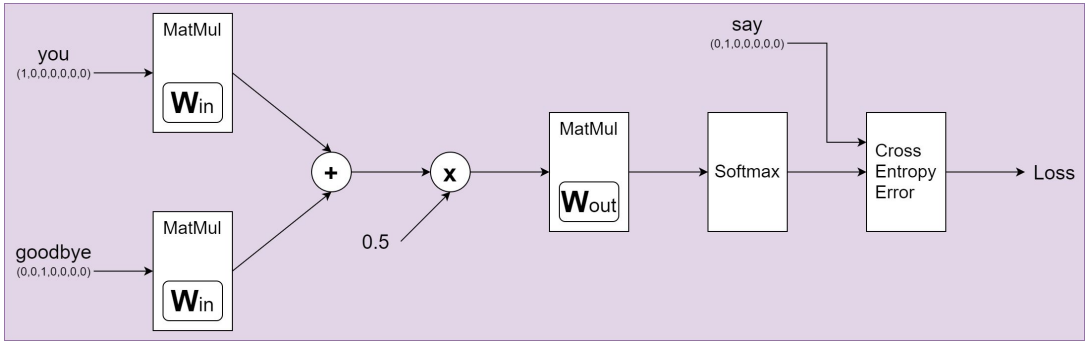
“say” は教師データと呼ばれる．学習においては，「？」に相当する語は予め分かっている

「？」を移動させていき，学習を逐次的に進めることで，重みがコーパスに最適化されていく

CBOW モデルの学習


you
(1,0,0,0,0,0,0)
MatMul
Win
goodbye
(0,0,1,0,0,0,0)
MatMul
Win
+
0.5
x
MatMul
Wout
Softmax
say
(0,1,0,0,0,0,0)
Cross
Entropy
Error
Loss

CBOW モデルの学習



損失の計算:

多クラス分類を行う NN

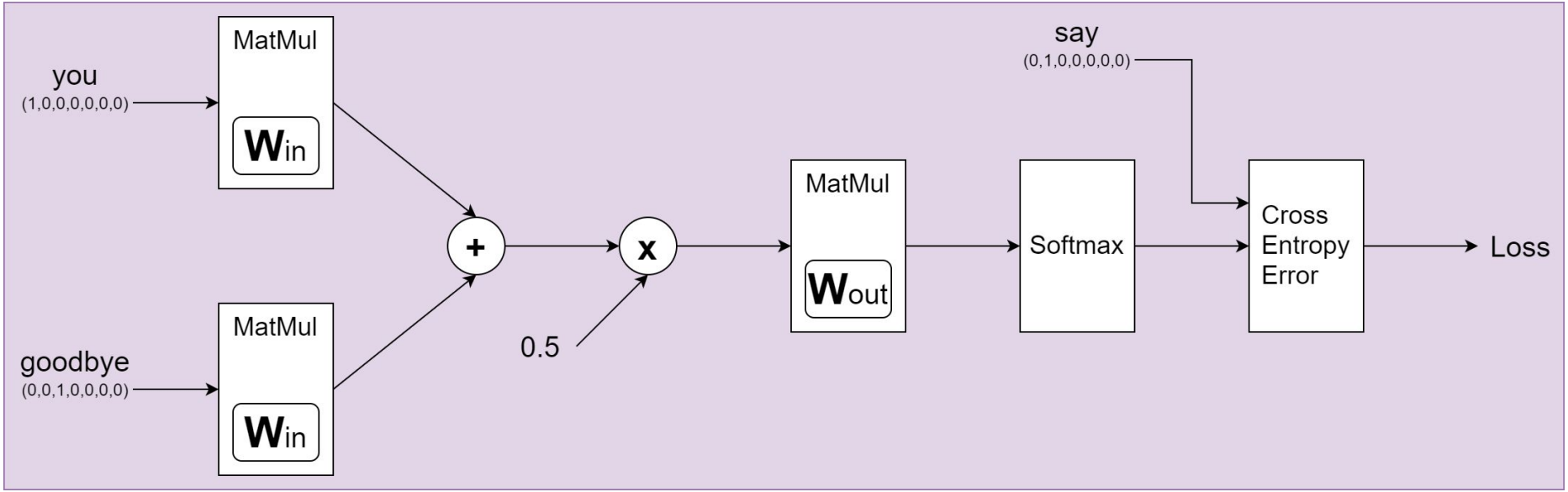
-> 学習には Softmax と交差エントロピー誤差を用いるだけ

スコアを Softmax で確率に変換し，その確率を教師ラベルから交差エントロピー誤差を求め，それを損失として学習を行う

交差エントロピー誤差とは？:

https://qiita.com/kenta1984/items/59a9ef1788e6934fd962

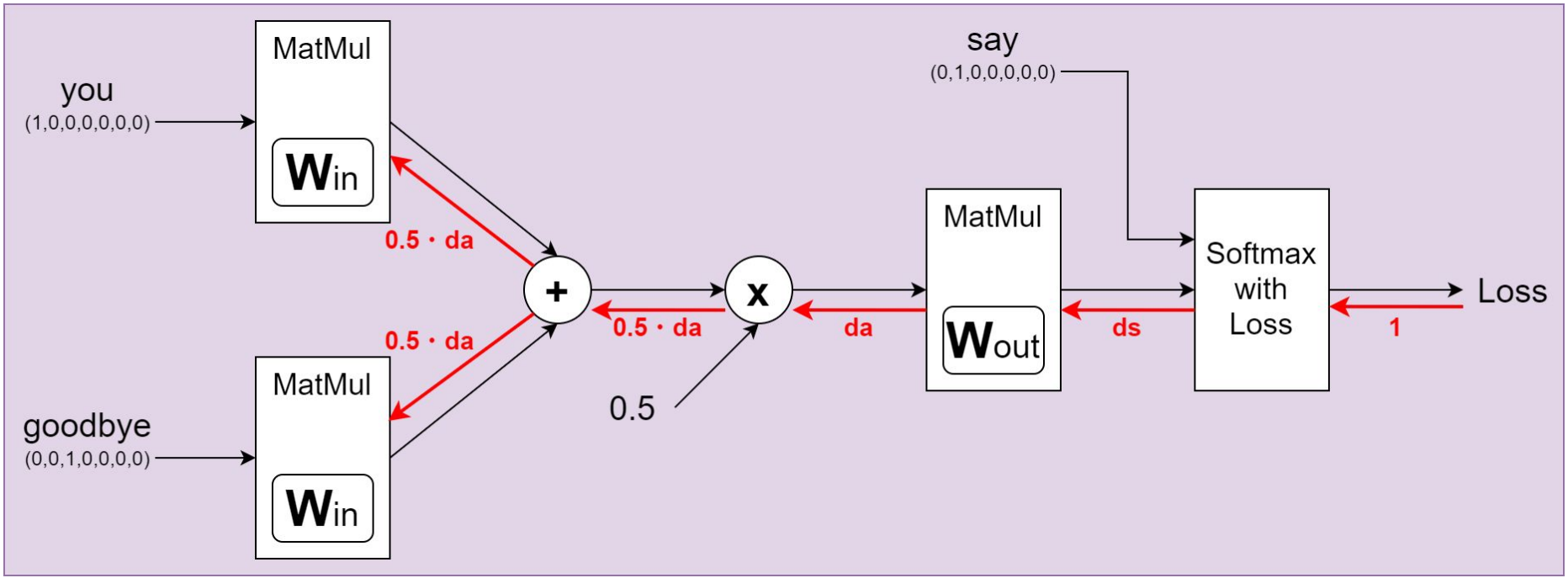
CBOW モデルの学習 -順伝播-



CBOW モデルの学習 -順伝播-

```
def forward(self, contexts, target):
        h0 = self.in_layer0.forward(contexts[:, 0])
        h1 = self.in_layer1.forward(contexts[:, 1])
        h = (h0 + h1) * 0.5
        score = self.out_layer.forward(h)
        loss = self.loss_layer.forward(score, target)
        return loss
```

CBOW モデルの学習 -逆伝播-



CBOW モデルの学習 -逆伝播-

```
def backward(self, dout=1):
        ds = self.loss_layer.backward(dout)
        da = self.out_layer.backward(ds)
        da *= 0.5
        self.in_layer1.backward(da)
        self.in_layer0.backward(da)
        return None
```

CBOW モデルの学習 -逆伝播-

勾配を順伝播とは逆方向に伝播する

「x」の逆伝播は，順伝播時の入力値を”入れ替えて”勾配に乗算

「＋」の逆伝播は，勾配を”そのまま通す”だけ

forward() メソッドを呼び，backward() メソッドを呼ぶことで，勾配が更新される

CBOW モデルの学習

あとは

ch03/train.py を見ましょう

カウントベース vs 推論ベース

word2vec は重みの再学習ができるため，単語の分散表現の更新や追加が効率的に行える

おまけ

word2vec 日本語モデル

word2vec 日本語学習済みモデル

https://aial.shiroyagi.co.jp/2017/02/japanese-word2vec-model-builder/

$pip install gensim

```
from gensim.models.word2vec import Word2Vec

model_path = '/path/to/word2vec.gensim.model'

model = Word2Vec.load(model_path)
```

形態素解析

形態素解析とは、私たちが普段生活の中で一般的に使っている言葉、つまり「自然言語」を形態素にまで分割する技術のことです。

形態素とは、言葉が意味を持つまとまりの単語の最小単位のことです。

日本語の形態素解析器は，juman++, mecab, janome などがある

juman++ が最も精度が高い，janome が導入が一番ラク

$pip install janome

問題

「りんごは赤い果物である」

この文を形態素に分け，それぞれの形態素の類似語を求めてみよう（類似語の数はいくつでも）

janome で形態素に分け，word2vec の most_similar() メソッドで類似語を検索

参考サイト：

https://note.nkmk.me/python-janome-tutorial/

https://qiita.com/kenta1984/items/93b64768494f971edf86

問題

「りんごは赤い果物である」

実行結果の例:

りんご
('みかん', 0.924630343914032)
は
('が', 0.6232839226722717)
赤い
('青い', 0.9212562441825867)
果物
('野菜', 0.9432708621025085)
で
('でも', 0.612108588218689)
ある
('あり', 0.8057374954223633)

WordNet の導入

$pip install nltk

対話モードで

>>> import nltk

>>> nltk.download('wordnet')

WordNet の詳細

https://www.nltk.org/howto/wordnet.html

WordNet

```
>>> from nltk.corpus import wordnet

>>> wordnet.synsets(‘car’) # car の同義語を取得

[Synset('car.n.01'), Synset('car.n.02'), Synset('car.n.03'), Synset('car.n.04'),
Synset('cable_car.n.01')]

>>> car = wordnet.synset(‘car.n.01’) # 同義語グループ

>>> car.definition()

'a motor vehicle with four wheels; usually propelled by an internal combustion
engine'
```

WordNet による意味の類似度

path_similarity() メソッド:

単語間の類似度を求める

```
>>> car = wordnet.synset(‘car.n.01’)
>>> novel = wordnet.synset(‘novel.n.01’)
>>> dog = wordnet.synset(‘dog.n.01’)
>>> motorcycle = wordnet.synset(‘motorcycle.n.01’)
>>> car.path_similarity(novel)
0.05555555555555555
```

問題

- cat の同義語(グループ)を取得し，それぞれの意味を列挙
  - synsets(), synset(), definition()

- dog と cat の類似度は？(どの同義語グループを比較してもよい)
  - synset(), path_similarlity()

- 前ページの4つの同義語グループを比較して，dog と最も類似している同義語グループは？

YOLO

画像認識 object detection

（もはや自然言語処理ではない）

python で動くdarkflow を使って画像認識をしてみよう

参考サイト:

https://qiita.com/watakandai/items/a7c9b908797aae5c6b60

付録

付録

本に載ってるコード集

https://github.com/oreilly-japan/deep-learning-from-scratch-2

> ch02, ch03 が今日やったところ